PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-056981
(43)Date of publication of application : 25.02.2000

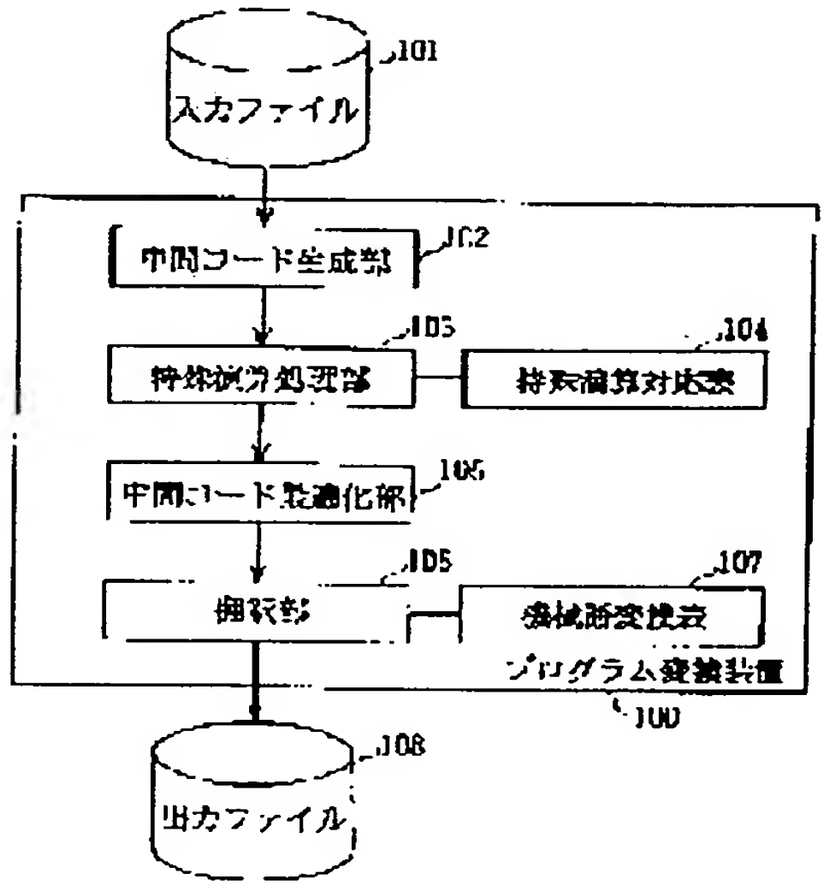
51)Int.Cl. G06F 9/45

21)Application number : 10-222657
22)Date of filing : 06.08.1998

(71)Applicant : MATSUSHITA ELECTRIC IND CO LTD
(72)Inventor : SAKATA TOSHIYUKI
TOMINAGA NOBUTERU
URUSHIBARA SEIICHI
HARUNA NAOSUKE

54) PROGRAM CONVERSION DEVICE

57)Abstract:
ROBLEM TO BE SOLVED: To provide a program conversion device converting a
ource program containing the description of effect for calling a DSP function formed of
machine word instruction string executing a product-sum operation and a saturation
peration into an efficient machine word instruction string.
OLUTION: An intermediate code generation part 102 converts description in the
ource program of effect for calling a DSP function into the intermediate code of
nction calling for calling the DSP function. A special operation processing part 103
onverts the intermediate code of function calling for calling the DSP function into an
termediate code for a special operation by referring to a special operation
rresponding table 104. An intermediate code optimizing part 105 sets the intermediate
de for the special operation to be the application object of optimization such as the
letion of a common part expression so that it is equal to the intermediate code of the
neration operation of a sum and a product. A translation part 106 converts the
termediate code optimized by the intermediate code optimizing part 105 into the
achine word instruction string and outputs it to an output file 108.



GAL STATUS
ate of request for examination]
ate of sending the examiner's decision of rejection]
ind of final disposal of application other than the examiner's
cision of rejection or application converted registration]
ate of final disposal for application]
atent number]
ate of registration]
umber of appeal against examiner's decision of rejection]
ate of requesting appeal against examiner's decision of rejection]
ate of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)**公開特許公報**（Ａ）　(11)特許出願公開番号
特開2000－56981
（P2000－56981A）
(43)公開日　平成12年２月25日(2000.2.25)

(51)Int.Cl.'	識別記号	ＦＩ	テーマコード'（参考）
Ｇ０６Ｆ　9/45		Ｇ０６Ｆ　9/44	３２２Ｆ　５Ｂ０８１

審査請求　未請求　請求項の数６　ＯＬ　（全 12 頁）

(21)出願番号　特願平10－222657

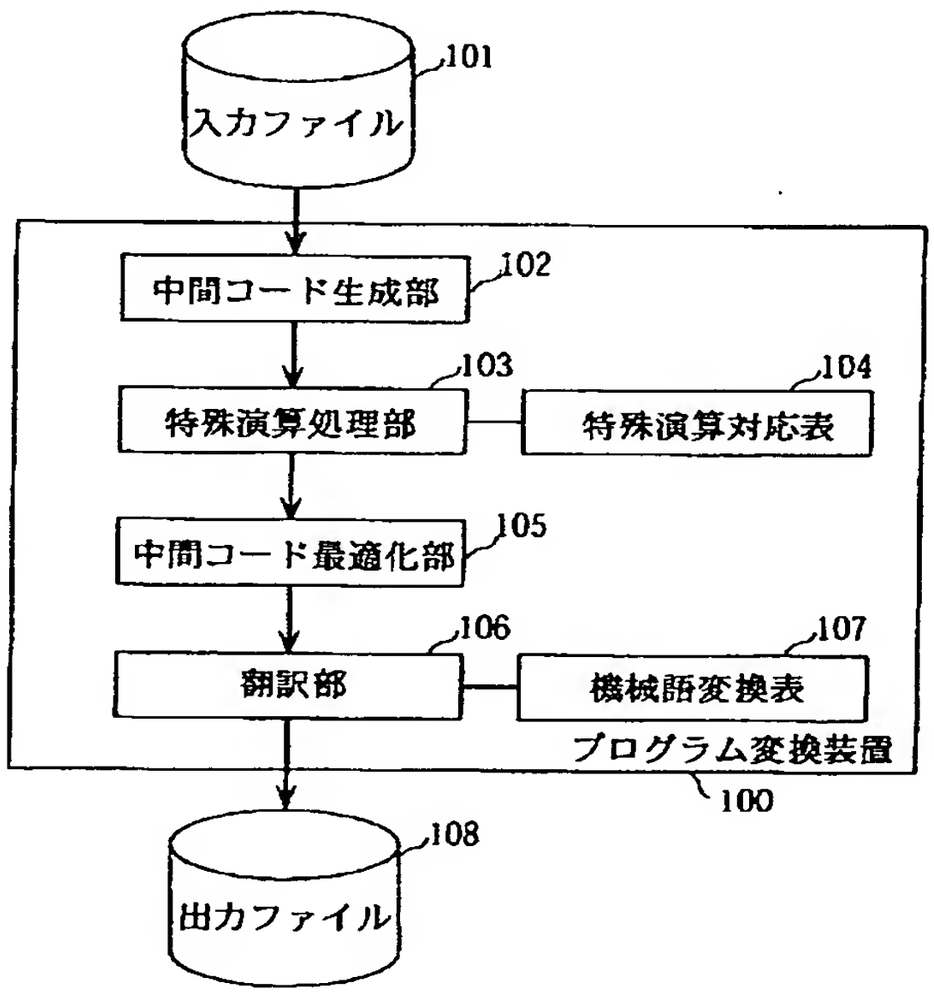
(22)出願日　平成10年８月６日(1998.8.6)

(71)出願人　000005821
松下電器産業株式会社
大阪府門真市大字門真1006番地
(72)発明者　坂田　俊幸
大阪府門真市大字門真1006番地　松下電器産業株式会社内
(72)発明者　富永　宣輝
大阪府門真市大字門真1006番地　松下電器産業株式会社内
(74)代理人　100090446
弁理士　中島　司朗　（外１名）

最終頁に続く

(54)【発明の名称】　プログラム変換装置

(57)【要約】
【課題】　積和演算、飽和演算等の演算を行う機械語命令列からなるＤＳＰ関数を呼び出す旨の記述を含むソースプログラムを、効率の良い機械語命令列に変換するプログラム変換装置を提供することを目的とする。
【解決手段】　ＤＳＰ関数を呼び出す旨のソースプログラム中の記述を、中間コード生成部１０２はＤＳＰ関数を呼び出す旨の関数呼出の中間コードに変換し、特殊演算対応表１０４を参照して特殊演算処理部１０３は、そのＤＳＰ関数を呼び出す旨の関数呼出の中間コードを特殊演算用の中間コードに変換する。この特殊演算用の中間コードを、中間コード最適化部１０５は、和や積等の一般的な演算の中間コードと同等に共通部分式の削除等の最適化の適用対象とする。中間コード最適化部１０５により最適化された中間コードを翻訳部１０６は機械語命令列に変換し出力ファイル１０８に出力する。



【特許請求の範囲】
【請求項１】　高級言語における算術演算の演算子とオペランドとからなる記述を適用対象とする最適化を行い高級言語で記述されたソースプログラムをプロセッサ用の機械語命令列に変換するプログラム変換装置であって、
引数を渡して所定関数名の関数を呼び出す旨の前記ソースプログラム中の特定関数呼出記述について最適化の適用の面において、当該引数をオペランドと同等に扱い、当該特定関数呼出記述を、オペランドと当該オペランドに算術演算を行う演算子との組と同等に扱うことにより、前記ソースプログラムを最適化された機械語命令列に変換することを特徴とするプログラム変換装置。
【請求項２】　前記プログラム変換装置は、
前記特定関数呼出記述を、最適化を行うに際し算術演算の演算子とオペランドとであると扱われるべき特性を有する中間コードに変換する変換手段と、
前記ソースプログラム中における前記特定関数呼出記述以外の部分を中間コードに変換する一般変換手段と、
前記変換手段及び前記一般変換手段による変換により生じた中間コードに基づいて最適化を行うことにより最適化された中間コードを生成する最適化手段と、
前記最適化手段により生成された中間コードを機械語命令列に変換する機械語変換手段とを備えることを特徴とする請求項１記載のプログラム変換装置。
【請求項３】　前記変換手段は、
引数を渡して所定の関数名の関数を呼び出す旨の情報を含む中間コードＡと、最適化を行うに際し算術演算の演算子とオペランドとであると扱われるべき特性を有する中間コードＢとを対応づけた対応関係情報を予め記憶する対応表記憶部と、
前記ソースプログラム中における関数を呼び出す旨の記述を、関数を呼び出す旨の情報と呼び出す関数の関数名を示す情報とを含む中間コードに変換する第１次変換部と、
前記対応表記憶部に記憶されている前記対応関係情報を参照し、前記第１次変換部による変換により生じた中間コードが、引数を渡して所定の関数名の関数を呼び出す旨の情報を含む中間コードＡであったときには、当該中間コードを、これに対応する中間コードＢに変換する第２次変換部とを有し、
前記最適化手段は、算術演算の演算子とオペランドとについての中間コードを最適化の適用対象とするものであり、算術演算の演算子とオペランドとであると扱われるべき特性を有する中間コードを、算術演算の演算子とオペランドとについての中間コードと同等な最適化の適用対象として最適化を行うことを特徴とする請求項２記載のプログラム変換装置。
【請求項４】　前記所定の関数名の関数は、積和演算及び飽和演算を含むディジタル信号処理用の演算のいずれかを実行することを処理内容とする関数であり、
前記プロセッサは、ディジタル信号処理用の演算命令を解読し実行することができるものであり、
前記機械語変換手段は、中間コードＢをディジタル信号処理用の演算命令に変換することを特徴とする請求項３記載のプログラム変換装置。
【請求項５】　前記最適化手段によりなされる最適化は、共通部分式の削除、定数伝搬、又はコピー伝搬を含むことを特徴とする請求項２～４のいずれか１項に記載のプログラム変換装置。
【請求項６】　高級言語における算術演算の演算子とオペランドとからなる記述を適用対象とする最適化を行い高級言語で記述されたソースプログラムをプロセッサ用の機械語命令列に変換するプログラム変換処理を、コンピュータに実行させるための制御プログラムを記録した記録媒体であって、
前記プログラム変換処理は、引数を渡して所定関数名の関数を呼び出す旨の前記ソースプログラム中の特定関数呼出記述について最適化の適用の面において、当該引数をオペランドと同等に扱い、当該特定関数呼出記述を、オペランドと当該オペランドに算術演算を行う演算子との組と同等に扱うことにより、前記ソースプログラムを最適化された機械語命令列に変換する処理であることを特徴とする記録媒体。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、高級言語で記述されたソースプログラムをプロセッサ用の機械語命令列に変換するプログラム変換装置に関し、特に積和演算・飽和演算等の特殊演算を行う関数を呼び出す記述を含むソースプログラムを変換対象とするプログラム変換装置に関する。
【０００２】
【従来の技術】近年、画像処理等のマルチメディア処理を高速に行なうために積和演算、飽和演算等のディジタル信号処理用の命令（以下、ＤＳＰ命令と呼ぶ。）がマイクロプロセッサに搭載されてきている。これらのマイクロプロセッサで用いられるソフトウェアは、通常、Ｃ言語、オブジェクト指向言語Ｃ＋＋（以下、単に「Ｃ＋＋言語」という。）等の高級言語で記述されたソースプログラムがプログラム変換装置で機械語命令に翻訳されることによって生成される。しかし、上述の高級言語はＤＳＰ命令を想定したものでないため、ＤＳＰ命令を示す演算子を備えていない。このため、従来、ＤＳＰ命令を使用するには、ＤＳＰ命令に対応する関数（以下、ＤＳＰ関数と呼ぶ。）を機械語命令で作成しておき、高級言語で記述されたソースプログラム中にそのＤＳＰ関数を呼び出す旨の記述をすることにしていた。なお、関数を呼び出す旨の記述は、高級言語で記述可能であり、標準化された高級言語で記述するソースプログラム中にＤ

ＳＰ関数を呼び出す旨の記述を用いることは、各種プロセッサに対応するソースプログラムを一本化できる等の利益をもたらす。

【０００３】以下、従来のプログラム変換装置が、図１に示すソースプログラムを、機械語命令列に変換する動作を説明する。図１は、ＤＳＰ関数を呼び出す旨の記述を含むソースプログラムの例を示す図である。同図に示すソースプログラムは、Ｃ言語で記述されたものであり、「＿ｓａｔａｄｄ（ｘ，ｙ）」がＤＳＰ関数を呼び出す旨の記述である。なお、このＤＳＰ関数は飽和加算演算を行う関数である。図１２は、従来のプログラム変換装置の機能ブロック図である。入力ファイル９０１には、ＤＳＰ関数を呼び出す旨の記述が含まれたソースプログラムが格納されている。プログラム変換装置９００は、入力ファイル９０１に格納されているソースプログラムを読み込み、機械語命令列に翻訳して、出力ファイル９０７に出力するいわゆるコンパイラであり、中間コード生成部９０３と、中間コード最適化部９０４と、翻訳部９０５と、ＤＳＰ関数展開部９０６とを備える。中間コード生成部９０３は、入力ファイル９０１から読み込みんだソースプログラムをプログラム変換装置内で使用する内部表現である中間コードに変換する。この中間コード生成部９０３が、図１のソースプログラムを変換した結果、図２に示す中間コードが生成される。図２は、図１に示すソースプログラムに対応してプログラム変換装置により生成される中間コードの例を示す図である。例えば、同図に示す中間コードにおける「ｔｍｐ＝＿ｓａｔａｄｄ（ｘ，ｙ）」、「ｔｍｐ　ｃｍｐｇｔ　０」が、図１に示すソースプログラムにおける「＿ｓａｔａｄｄ（ｘ，ｙ）＞０」に対応している。

【０００４】中間コード最適化部９０４は、中間コード生成部９０３で生成した中間コードに対して、関数のインライン展開、共通部分式の削除、定数伝搬、コピー伝搬等の最適化を行なう。但し、図２に示す中間コード中の「＿ｓａｔａｄｄ（ｘ，ｙ）」は、ＤＳＰ関数の呼出の意味を持つため、複数含まれているにもかかわらず共通部分式とは扱えず、中間コード最適化部９０４は、共通部分式の削除を行わない。ここで、関数のインライン展開は、関数を呼び出している箇所を関数内容を示す中間コードに置き換えることであり、これによりサブルーチンリンケージ及びそれに伴う命令が不要になる。共通部分式の削除は、部分式が等価な値をもつときに、その評価を１回行うとその結果を変数に格納し、その後の共通な部分式をその変数で置き換えることであり、これにより複数の等価な部分式の評価を１回で済ませることができる。定数伝搬は、例えば、定数ａを変数ｖに代入するｖ＝ａという代入文以後、変数ｖへの代入が現れるまでｖへの参照を定数ａで置き換えることであり、これにより、式中の変数が定数に置き換えられる場合にはその部分を翻訳時に計算して、もとの式を計算結果に置き換えること等が可能になる。また、コピー伝搬は、定数伝搬における定数を変数に代えたものである。なお、高級言語で記述されたソースプログラムにおける「ｘ＋ｙ」や「ｘ－ｙ」等の演算子とオペランドは、中間コードに変換された後も、共通部分式の削除、定数伝搬、コピー伝搬等の最適化を適用する対象となる。翻訳部９０５は、中間コード最適化部９０４で最適化した中間コードを機械語命令に翻訳する。この翻訳結果を図１３に示す。

【０００５】図１３は、従来のプログラム変換装置９００における翻訳部９０５の出力の例を示す図である。ＤＳＰ関数展開部９０６は、翻訳部９０５で翻訳した機械語命令の中でＤＳＰ関数を呼び出すサブルーチンコール命令を検出し、これを、機械語ＤＳＰ関数格納ファイル９０２に格納されている機械語命令列であるＤＳＰ関数に展開し、出力ファイル９０７に結果を出力する。ＤＳＰ関数展開部９０６の処理結果を図１４に示す。図１４は、従来のプログラム変換装置９００におけるＤＳＰ関数展開部９０６の処理結果の例を示す図である。なお、図中においては、機械語命令列は、便宜上、アセンブリ言語のプログラムとして表現している。このように、従来のプログラム変換装置９００は、図１に示すソースプログラムの一部を図１４に示す機械語命令列に変換する。

【０００６】

【発明が解決しようとする課題】従来のプログラム変換装置は、ＤＳＰ関数が機械語命令で記述されており中間コード段階ではインライン展開ができないため、ソースプログラムが機械語命令に翻訳された後にＤＳＰ関数のインライン展開を行う（図１３、図１４参照）。このため、ソースプログラムにおけるＤＳＰ関数を呼び出す旨の記述については、中間コードに変換した後も中間コード最適化部９０４における最適化の対象とならないため、ソースプログラムを効率の良い機械語命令列に変換することができないという問題がある。そこで、本発明は、このような問題点に鑑みてなされたものであって、ＤＳＰ関数を呼び出す旨の記述を含むソースプログラムを、効率の良い機械語命令列に変換するプログラム変換装置を提供することを目的とする。

【０００７】

【課題を解決するための手段】上記課題を解決するために、本発明に係るプログラム変換装置は、高級言語における算術演算の演算子とオペランドとからなる記述を適用対象とする最適化を行い高級言語で記述されたソースプログラムをプロセッサ用の機械語命令列に変換するプログラム変換装置であって、引数を渡して所定関数名の関数を呼び出す旨の前記ソースプログラム中の特定関数呼出記述について最適化の適用の面において、当該引数をオペランドと同等に扱い、当該特定関数呼出記述を、オペランドと当該オペランドに算術演算を行う演算子と

の組と同等に扱うことにより、前記ソースプログラムを最適化された機械語命令列に変換することを特徴とする。即ち、本発明は、特殊演算を行う機械語命令列からなる関数を呼び出す旨の特定関数呼出記述が、関数呼出の形式をとりつつも、その内容は算術演算であることに着目したものであり、本発明に係るプログラム変換装置は、特定関数呼出記述を和や積等の通常の演算と同等に共通部分式削除等の最適化の適用対象とするので、特定関数呼出記述を含むソースプログラムを実行効率の良い機械語命令列に変換することができる。

【０００８】

【発明の実施の形態】以下、本発明の実施の形態に係るプログラム変換装置について、図面を用いて説明する。

＜構成＞図３は、本発明の実施の形態に係るプログラム変換装置１００の機能ブロック図である。なお、図３には入力ファイル１０１、出力ファイル１０８も併せて記載している。ここで、入力ファイル１０１は、ＤＳＰ関数を呼び出す旨の記述を含むＣ言語のソースプログラムを格納している。また、出力ファイル１０８は、プログラム変換装置１００により、入力ファイル１０１に格納されたソースプログラムの翻訳結果である機械語命令列が格納されるファイルである。プログラム変換装置１００は、コンピュータに備えられたメモリに格納されたプログラムがＣＰＵにより実行されることで、ソースプログラムを機械語命令列に翻訳（変換）する動作を行ういわゆるコンパイラであり、機能的には、中間コード生成部１０２、特殊演算処理部１０３、特殊演算対応表１０４、中間コード最適化部１０５、翻訳部１０６及び機械語変換表１０７から構成される。

【０００９】ここで、ソースプログラムを機械語命令列に翻訳するとは、ソースプログラムに基づいて、当該ソースプログラムに記述された処理内容をプロセッサに実行させるための機械語命令列を生成することをいう。中間コード生成部１０２は、入力ファイル１０１に格納されているソースプログラムをメモリに読み込み、このソースプログラムをプログラム変換装置内で使用する内部表現である中間コードに変換する。即ち、中間コード生成部１０２は、ソースプログラムに基づいて中間コードを生成する。なお、中間コード生成部１０２は、上述した従来のプログラム変換装置９００における中間コード生成部９０３と同等である。特殊演算処理部１０３は、後述する特殊演算対応表１０４を参照することにより、ＤＳＰ関数呼出の中間コードの変換処理を行う。即ち、特殊演算処理部１０３は、中間コード生成部１０２により生成された中間コードの中から、ＤＳＰ関数を呼び出す旨の中間コードを検出して、これを特殊演算用の中間コードに変換する。中間コード最適化部１０５は、特殊演算処理部１０３の処理結果である中間コードに対して、関数のインライン展開、共通部分式の削除、定数伝搬、コピー伝搬等の最適化を行なう。翻訳部１０６は、

後述する機械語変換表１０７を参照することにより、中間コード最適化部１０５によって最適化された中間コードを機械語命令に翻訳し、出力ファイル１０８に出力する。

【００１０】＜特殊演算対応表＞図４は、特殊演算対応表１０４の内容を示す図である。特殊演算対応表１０４には、ＤＳＰ関数を呼び出す旨の中間コードと、それに対応する特殊演算用の中間コードとの組が複数記憶されている。同図中の「＿ｓａｔａｄｄ（ｘ，ｙ）」等のようなＤＳＰ関数を呼び出す旨の中間コードは、一般の関数を呼び出す旨の中間コードと形式上は同一である。特殊演算処理部１０３は、関数呼出の中間コードを、特殊演算対応表１０４に当該中間コードが記憶されているか否かにより、ＤＳＰ関数を呼び出す旨の中間コードであるか一般の関数を呼び出す旨の中間コードであるかを判断する。この特殊演算対応表１０４に記憶されているＤＳＰ関数に関する情報は、ソースプログラム作成者に認識されている必要がある。但し、このＤＳＰ関数に関する情報を共通して備えるソースプログラム開発支援ツール等によって、プログラム作成者の認識負担を軽減する等していてもよい。ここでは、ソースプログラムにおいてＤＳＰ命令を使う場合は、特殊演算対応表１０４に記憶されている関数名の関数を呼び出す旨の記述を行うという取り決めに従ってソースプログラムが作成されているものとする。同図中の「ｘ　ｓａｔａｄｄ　ｙ」等のような特殊演算用の中間コードは、従来のプログラム変換装置では用いられていなかった新たな中間コードである。ここで、「ｓａｔａｄｄ」は飽和加算演算を示し、例えば、ｘとｙの和を表す一般的な中間コード「ｘ　ａｄｄ　ｙ」における「ａｄｄ」のように、演算子の意味を有する中間コードである。また、「ｘ　ｓａｔａｄｄ　ｙ」における「ｘ」と「ｙ」とは演算子に対するオペランドの意味を有する。なお、「ｘ　ａｄｄｙ」はソースプログラムにおける「ｘ＋ｙ」を中間コードで表現したものである。また、「ｓａｔｓｕｂ」は飽和減算演算を示し、「ｓａｔｍｕｌ」は飽和乗算演算を示している。

【００１１】このように、特殊演算用の中間コードは、演算子とオペランドとの意味を有する中間コードであるため、その意味が中間コード最適化部１０５により参照され、最適化の適用の面において、高級言語における通常の演算子を中間コードに変換したものと同等に扱われる。なお、特殊演算対応表１０４におけるｘ、ｙ等は、変数名そのものを特定したものではなく、それぞれ第１引数、第２引数を表現したものである。従って、引数は、いかなる変数名であっても、いかなる値のものであってもよい。例えば、この特殊演算対応表１０４に従えば、「＿ｓａｔａｄｄ（ｙ，２）」を「ｙ　ｓａｔａｄｄ　２」に変換することができる。また、中間コード段階での最適化において、演算子及びオペランドからなる

式については共通部分式の削除の対象となり得、また、オペランドについては定数伝搬、コピー伝搬の対象となり得る。

【００１２】＜機械語変換表＞図５は、機械語変換表１０７の内容を示す図である。なお、同図中においては、機械語命令は、便宜上、ニモニックコードで表現している。機械語変換表１０７には、中間コードと、それに対応する１又は複数の機械語命令との組が複数記憶されている。例えば、「ｓａｔａｄｄ」という特殊演算用の中間コードは、「ｓａｄｄ」という飽和加算命令と対応づけられており、「ｓａｔｓｕｂ」という特殊演算用の中間コードは、「ｓｓｕｂ」という飽和減算命令と対応づけられている。ここで、機械語命令である「ｓａｄｄ」や「ｓｓｕｂ」は、プログラム変換装置１００の対象とするプロセッサが解読実行可能な命令である。この機械語変換表１０７において、特殊演算用の中間コードと機械語命令との対応関係以外の情報は、一般のプログラム変換装置において中間コードから機械語命令に変換するために用いられる情報と同様である。

【００１３】＜動作＞以下、上述の構成を備えるプログラム変換装置１００の動作について、図１に示すソースプログラムを変換対象とした場合を例にして説明する。図６は、プログラム変換装置１００の動作を示すフローチャートである。中間コード生成部１０２は、入力ファイル１０１に格納されているソースプログラム（図１参照）をメモリに読み込み（ステップＳ２１０）、ソースプログラムに基づき中間コードをメモリ中に生成する（ステップＳ２２０）。生成された中間コードは図２に示すものとなる。図２に示す中間コードの意味は以下の通りである。「ｔｍｐ＝＿ｓａｔａｄｄ（ｘ，ｙ）」は、変数ｘとｙの値を引数として、関数名が＿ｓａｔａｄｄである関数の呼び出しを行ない、結果を一時変数ｔｍｐに代入する中間コードである。「ｔｍｐ　ｃｍｐｇｔ　０」は、一時変数ｔｍｐが定数０より大きいか否かを判定する中間コードである。「ｊｍｐ＿ｆａｌｓｅ　Ｌａｂｅｌｌ」は、判定結果が偽ならラベルＬａｂｅｌｌに分岐する条件分岐の中間コードである。「ａ＝＿ｓａｔａｄｄ（ｘ，ｙ）」は、変数ｘとｙの値を引数として、関数＿ｓａｔａｄｄの呼び出しを行ない、結果を変数ａに代入する中間コードである。「Ｌａｂｅｌｌ」は、分岐先のラベルを示す中間コードである。中間コード生成部１０２により中間コードが生成された後、特殊演算処理部１０３は、ＤＳＰ関数呼出の中間コードの変換処理を行う（ステップＳ２３０）。

【００１４】以下、図７及び図８を用いてＤＳＰ関数呼出の中間コードの変換処理について詳細に説明する。図７は、特殊演算処理部１０３の行うＤＳＰ関数呼出の中間コードの変換処理を示すフローチャートである。特殊演算処理部１０３は、中間コード生成部１０２により生成された中間コード（図２参照）の中から、関数呼出の中間コードを検索する（ステップＳ２３１）。関数呼出の中間コードが検出できなければ（ステップＳ２３２）、ＤＳＰ関数呼出の中間コードの変換処理は終了し、関数呼出の中間コードが検出できれば（ステップＳ２３２）、特殊演算処理部１０３は、それがＤＳＰ関数呼出の中間コードであるか判定する（ステップＳ２３３）。関数呼出の中間コードが、ＤＳＰ関数呼出の中間コードであるかの判定は（ステップＳ２３３）、その関数呼出の中間コードが特殊演算対応表１０４に記憶されている中間コードであるか否かによって行う。例えば、図２に示す中間コードのうちの関数呼出の中間コード「＿ｓａｔａｄｄ（ｘ，ｙ）」は、特殊演算対応表１０４に記憶されているものであるため（図４参照）、特殊演算処理部１０３により、ＤＳＰ関数呼出の中間コードであると判定される。ステップＳ２３３における判定の結果、ステップＳ２３２において検出した関数呼出の中間コードはＤＳＰ関数呼出の中間コードでないと判定した場合には、特殊演算処理部１０３は、ステップＳ２３１に戻り、中間コード生成部１０２により生成された中間コードの中から他の関数呼出の中間コードの検索を行う。また、ステップＳ２３３における判定の結果、ＤＳＰ関数呼出の中間コードであると判定した場合には、特殊演算処理部１０３は、特殊演算対応表１０４を参照することにより、ＤＳＰ関数呼出の中間コードと対応する特殊演算用の中間コードを得て（図４参照）、これを生成し、ＤＳＰ関数呼出の中間コードと置き換え（ステップＳ２３４）、ステップＳ２３１の処理に戻る。例えば、「＿ｓａｔａｄｄ（ｘ，ｙ）」は、「ｘ　ｓａｔａｄｄ　ｙ」に置き換えられる。

【００１５】このような、特殊演算処理部１０３によるＤＳＰ関数呼出の中間コードの変換処理がなされた結果として、図２に示す中間コードは、図８に示す中間コードに変換される。図８は、特殊演算処理部１０３により変換された後の中間コードの例を示す図である。同図に示すように、特殊演算処理部１０３によって、図２に示す中間コード「ｔｍｐ＝＿ｓａｔａｄｄ（ｘ，ｙ）」、「ａ＝＿ｓａｔａｄｄ（ｘ，ｙ）」は、それぞれ「ｔｍｐ＝ｘ　ｓａｔａｄｄ　ｙ」、「ａ＝ｘ　ｓａｔａｄｄ　ｙ」に置き換えられている。即ち、引数を渡して関数＿ｓａｔａｄｄを呼び出す旨の情報を有する関数呼出の中間コードが、ｓａｔａｄｄという演算子とオペランドとである旨の情報を有する特殊演算用の中間コードに置き換えられたのである。以下、再び、図６のフローチャートに基づく説明に戻る。特殊演算処理部１０３によりＤＳＰ関数呼出の中間コードの変換処理（ステップＳ２３０）がなされたメモリ中の中間コードに対して、中間コード最適化部１０５は、最適化を行う（ステップＳ２４０）。即ち、中間コード最適化部１０５は、関数のインライン展開、共通部分式の削除、定数伝搬、コピー伝搬等の最適化を行なう。ここで、図８に示す中間コード

「ｘ　ｓａｔａｄｄ　ｙ」における「ｓａｔａｄｄ」等の演算子の意味を有する中間コードは、共通部分式の削除の対象となり、中間コード最適化部１０５は、「ａ＝ｘ　ｓａｔａｄｄ　ｙ」を「ａ＝ｔｍｐ」に置き換える。なお、オペランドの意味を有する中間コードは、定数伝搬、コピー伝搬の対象となる。また、最適化の過程において、飽和加算等の特殊演算用の中間コードは、そのオペランドが定数となれば、その特殊演算用の中間コードによる演算を計算してしまい、もとの特殊演算用の中間コードをその計算結果で置き換えることができる。「ｘ　ｓａｔａｄｄ　ｙ」は、高級言語における「ｘ＋ｙ」を変換した中間コード「ｘ　ａｄｄ　ｙ」と、最適化の適用対象という面においては全く同等に扱われる。

【００１６】中間コード最適化部１０５による処理結果を図９に示す。図９は、図８に示す中間コードに対する中間コード最適化部１０５による処理結果を示す図である。中間コード最適化部１０５により最適化がなされた後、翻訳部１０６は、機械語変換表１０７を参照しつつ中間コードを機械語命令列に変換し（ステップＳ２５０）、機械語命令列を出力ファイル１０８に出力する（ステップＳ２６０）。例えば、特殊演算処理部１０３によって生成された「ｓａｔａｄｄ」という中間コードは、翻訳部１０６によって「ｓａｄｄ」という飽和加算命令に変換される。図１０は、図９に示す中間コードを翻訳部１０６で機械語命令列に変換した結果を示す図である。なお、機械語命令列は、便宜上、アセンブリ言語のプログラムとして表現している。図１０に示す機械語命令の意味は以下の通りである。「ｍｏｖ　（ｘ），Ｒ３」は、変数ｘの値をレジスタＲ３にロードする。「ｍｏｖ　（ｙ），Ｒ４」は、変数ｙの値をレジスタＲ４にロードする。「ｓａｄｄ　Ｒ３，Ｒ４」は、レジスタＲ３の値とレジスタＲ４の値を飽和加算し、結果をレジスタＲ４に格納する。「ｃｍｐ　Ｒ４，０」は、レジスタＲ４の値と定数０を比較する。「ｂｌｔ　Ｌａｂｌ」は、比較結果により条件分岐する。「ｍｏｖ　Ｒ４，（ａ）」は、レジスタＲ４の値を変数ａにストアする。「Ｌａｂｌ：」は、分岐命令の飛び先ラベルである。このように、プログラム変換装置１００は、ソースプログラム中のＤＳＰ関数を呼び出す旨の記述を、関数呼出の中間コードに変換した後に、最適化の面で演算子を示す中間コードと同様に扱える形式の特殊演算命令用の中間コードに変換し、その後、中間コード段階で最適化を行って、機械語命令列に翻訳する。

【００１７】＜考察＞本発明の実施の形態に係るプログラム変換装置１００の処理結果（図１０）を、従来のプログラム変換装置９００の処理結果（図１４）と比較して説明する。両者の前提となる状態は、図２に示す中間コードである。プログラム変換装置１００は、「ｓａｄｄ」を１つ生成しているが（図１０参照）、従来のプログラム変換装置９００は、「ｓａｄｄ」を２つ生成している（図１４参照）。この差異は、関数呼び出しの中間コード「＿ｓａｔａｄｄ（ｘ，ｙ）」を、プログラム変換装置９００を共通部分式の削除という最適化の対象としないが、プログラム変換装置１００では共通部分式の削除という最適化の対象とすることによるものである。また、従来のプログラム変換装置９００の処理結果には、「ｍｏｖ　Ｒ１，Ｒ３」、「ｍｏｖ　Ｒ２，Ｒ４」というレジスタＲ１及びＲ２の待避のためのコードが含まれている。これは、従来のプログラム変換装置９００が、「＿ｓａｔａｄｄ（ｘ，ｙ）」を機械語命令列に翻訳する場合に、関数呼び出しにおける引数受渡しで使用するレジスタは常にＲ１及びＲ２と決められているために、レジスタＲ１及びＲ２がすでに使用されているときには、関数呼び出しの前にレジスタＲ１及びＲ２を退避するコードを生成してしまうことによる。これに対し、プログラム変換装置１００では、図２に示す「＿ｓａｔａｄｄ（ｘ，ｙ）」を中間コード段階で「ｘ　ｓａｔａｄｄ　ｙ」に変換した後に機械語命令に翻訳するので、関数呼出に関連したレジスタについての制約を受けず、レジスタＲ３及びＲ４を使用することができ、このためレジスタＲ１及びＲ２の待避等のコードを生成しない。このように、図１０、図１４に示す機械語命令列を比較すれば、本発明に係るプログラム変換装置が生成した機械語命令列の方が、従来のプログラム変換装置が生成した機械語命令列よりも、効率の良いものであることは明らかである。

【００１８】以上、本発明に係るプログラム変換装置について、実施形態に基づいて説明したが、本発明はこれら実施形態に限られないことは勿論である。即ち、

（１）実施の形態では、プログラム変換装置が対象とするプロセッサの命令セットに「ｓａｄｄ」という飽和加算命令が含まれていることを前提として、特殊演算処理部１０３によって生成された「ｓａｔａｄｄ」という中間コードは、翻訳部１０６によって「ｓａｄｄ」という飽和加算命令に変換されることとしたが、プロセッサの命令セットに飽和加算命令等のＤＳＰ命令が含まれていなくてもよい。この場合には、機械語変換表１０７には、飽和加算命令と等価な動作を実現するような複数の機械語命令と、「ｓａｔａｄｄ」という中間コードとを対応づけておき、翻訳部１０６はこれを参照して「ｓａｔａｄｄ」を複数の機械語命令列に変換することになる。

【００１９】（２）実施の形態では、プログラム変換装置は、ソースプログラムを機械語命令列に翻訳することとしたが、この機械語命令列に翻訳とは、これと同等のアセンブリ言語プログラムに翻訳することを含むものであり、プログラム変換装置がアセンブリ言語プログラムを出力するものであってもよい。この場合は、このアセンブリ言語のプログラムはいわゆるアセンブラオプティマイザ等に入力され、さらに最適化された機械語命令列

に変換され得る。

【００２０】（３）実施の形態では、特殊演算処理部１０３によって生成された「ｘ　ｓａｔａｄｄ　ｙ」等の特殊演算用の中間コードは、演算子とオペランドとの意味を有する中間コードであるため、その意味が中間コード最適化部１０５により参照され、最適化の適用の面において、高級言語における通常の演算子を中間コードに変換したものと同等に扱われることとしたが、中間コードに演算子とオペランドの意味をもたせる方法は、一定書式の中間コードを用いることに限定されることはない。即ち、中間コード最適化部１０５が演算子である中間コードを列挙した表を参照することにより、最適化の適用の可否を判断するものであるならば、この表に特殊演算用の中間コードを追加するような方法であってもよい。また、中間コード最適化部１０５が特殊演算対応表１０４を参照することにより、特殊演算対応表１０４に含まれている中間コードを演算子の中間コードと同等に扱うことにしてもよい。

【００２１】（４）実施の形態では、特殊演算対応表１０４は、予め完成したものとしてプログラム変換装置１００に備えられているとしたが、特殊演算対応表１０４は、外部のファイルに格納された所定の情報に基づきプログラム変換装置１００によって生成されることとしてもよい。従って、実施の形態では、ソースプログラムにおいてＤＳＰ命令を使う場合は、特殊演算対応表１０４に記憶されている関数名の関数を呼び出す旨の記述を行うという取り決めに従ってソースプログラムが作成されているものという前提をおいたが、逆に、ソースプログラム中でＤＳＰ命令を使うために、引数を渡してある関数名の関数を呼び出す旨の記述をしたならば、その引数と関数名とに関する情報をファイルに格納してプログラム変換装置１００に供給することとしてもよい。この場合、供給されたファイルを参照してプログラム変換装置１００は、特殊演算対応表１０４を作成する。

【００２２】（５）実施の形態における機械語変換表１０７には、例えば、「ｓａｔａｄｄ」という特殊演算用の中間コードが「ｓａｄｄ」という飽和加算命令と対応づけられているように、中間コードと機械語命令とが対応づけられていることとしたが、従来のプログラム変換装置と同等レベルの一般的な中間コードと機械語命令との対応関係を除く、特殊演算用の中間コードと機械語命令との対応関係に関する情報は、外部のファイルに格納された所定の情報に基づきプログラム変換装置１００によって生成されることとしてもよい。例えば、ＤＳＰ関数を呼び出す旨のソースプログラム中の記述を手掛かりとし、ＤＳＰ関数の本体である機械語プログラムモジュールが格納されているファイルを参照することにより、当該ＤＳＰ関数に関する中間コードの変換結果となる機械語命令を特定することとしてもよい。また、プログラム変換装置１００は、機械語変換表１０７を複数備えて切り替えて用いることにすれば、複数の異なるプロセッサに対応する機械語命令列を出力することができる。

【００２３】（６）実施の形態では、特殊演算処理を行なうＤＳＰ関数を一般の関数と同様の呼出記述をもって記述したソースプログラムを変換対象としたが、ＤＳＰ関数が、Ｃ＋＋言語の多重定義演算子として定義されているようなソースプログラムを変換対象とすることもできる。多重定義演算子は本質的に関数と同等であり、第１次的には関数を呼び出す旨の中間コードに変換されるものだからである。なお、多重定義演算子を用いるとプログラムの可読性は高まる。ちなみに、図１に示したソースプログラムをＣ＋＋言語の多重定義演算子を用いて記述した例を図１１に示す。

【００２４】（７）実施の形態では、中間コードが演算子又はオペランドの意味を有するものであれば、中間コード最適化部１０５はその中間コードを共通部分式の削除等の最適化の対象とすることとしたが、この中間コード段階における最適化と同様のことを、中間コードを機械語命令列に翻訳した後に、中間コードに関する全ての情報を利用することにより行うこととしてもよい。

【００２５】（８）実施の形態におけるプログラム変換装置の処理手順（図６、図７のフローチャートの手順等）を機械語プログラムにより実現し、これを記録媒体に記録して流通・販売の対象にしても良い。このような記録媒体には、ＩＣカード、光ディスク、フレキシブルディスク、ＲＯＭ等があるが、これらに記録された機械語プログラムは、汎用のコンピュータにインストールされることにより利用に供される。即ち、汎用のコンピュータは、インストールした上記機械語プログラムを逐次実行して、実施の形態に示したようなプログラム変換装置を実現する。また、汎用のコンピュータに上述のプログラム変換装置の処理手順を実行させるためのコンピュータプログラムは、ハードディスク等の記録媒体及び通信回線等を介してオンラインで流通させ頒布することもできる。

【００２６】

【発明の効果】以上の説明から明らかなように、本発明に係るプログラム変換装置は、高級言語における算術演算の演算子とオペランドとからなる記述を適用対象とする最適化を行い高級言語で記述されたソースプログラムをプロセッサ用の機械語命令列に変換するプログラム変換装置であって、引数を渡して所定関数名の関数を呼び出す旨の前記ソースプログラム中の特定関数呼出記述について最適化の適用の面において、当該引数をオペランドと同等に扱い、当該特定関数呼出記述を、オペランドと当該オペランドに算術演算を行う演算子との組と同等に扱うことにより、前記ソースプログラムを最適化された機械語命令列に変換することを特徴とする。これにより、引数を渡してＤＳＰ関数を呼び出す旨のソースプログラム中の記述を、単なる関数呼出としてではなく、高

級言語で記述可能な「ｘ＋ｙ」等の演算子及びオペランドと同等に最適化適用対象とするため、ＤＳＰ関数の処理内容が演算であるという特質を生かした最適化が行える。従って、本発明に係るプログラム変換装置は、ＤＳＰ関数を呼び出す旨の記述を通常の関数呼出と同様に扱う従来のプログラム変換装置と比べて、ＤＳＰ関数を呼び出す旨の記述を含むソースプログラムを実行効率の良い機械語命令列に変換することができる。

【００２７】ここで、前記プログラム変換装置は、前記特定関数呼出記述を、最適化を行うに際し算術演算の演算子とオペランドとであると扱われるべき特性を有する中間コードに変換する変換手段と、前記ソースプログラム中における前記特定関数呼出記述以外の部分を中間コードに変換する一般変換手段と、前記変換手段及び前記一般変換手段による変換により生じた中間コードに基づいて最適化を行うことにより最適化された中間コードを生成する最適化手段と、前記最適化手段により生成された中間コードを機械語命令列に変換する機械語変換手段とを備えることとすることができる。これにより、引数を渡してＤＳＰ関数を呼び出す旨のソースプログラム中の記述に基づいて、演算子とオペランドとを意味する情報を有する特殊演算用の中間コードを生成するので、演算子又はオペランドであることに基づいて適用することが可能となる中間コード段階での最適化が、特殊演算用の中間コードに適用することができる。従って、本発明に係るプログラム変換装置は、従来のプログラム変換装置と比べて、ＤＳＰ関数を呼び出す旨の記述を含むソースプログラムを、一層最適化された中間コードに変換することができる。また、一般に、関数呼出の中間コードを機械語命令に翻訳する場合、引数及び戻り値の受渡しに使用するレジスタが予め決められている。このため、ＤＳＰ関数呼出の中間コードのままでは、機械語命令に翻訳する段階で、レジスタ割り付けに制限が生じてしまうが、本発明に係るプログラム変換装置では、この関数呼出の中間コードを特殊演算用の中間コードに変換するため、機械語命令への翻訳段階でもレジスタ割り付けの制限を受けず最適なレジスタ割り付けが行える。

【００２８】また、前記変換手段は、引数を渡して所定の関数名の関数を呼び出す旨の情報を含む中間コードＡと、最適化を行うに際し算術演算の演算子とオペランドとであると扱われるべき特性を有する中間コードＢとを対応づけた対応関係情報を予め記憶する対応表記憶部と、前記ソースプログラム中における関数を呼び出す旨の記述を、関数を呼び出す旨の情報と呼び出す関数の関数名を示す情報とを含む中間コードに変換する第１次変換部と、前記対応表記憶部に記憶されている前記対応関係情報を参照し、前記第１次変換部による変換により生じた中間コードが、引数を渡して所定の関数名の関数を呼び出す旨の情報を含む中間コードＡであったときには、当該中間コードを、これに対応する中間コードＢに

変換する第２次変換部とを有し、前記最適化手段は、算術演算の演算子とオペランドとについての中間コードを最適化の適用対象とするものであり、算術演算の演算子とオペランドとであると扱われるべき特性を有する中間コードを、算術演算の演算子とオペランドとについての中間コードと同等な最適化の適用対象として最適化を行うこととすることもできる。これにより、本発明に係るプログラム変換装置は、ＤＳＰ関数を呼び出す旨の情報を含む中間コードとこれを変換した後の中間コードとの対応関係を予め記憶し、これを参照して、中間コードの変換を行うものであるため、この記憶内容を追加更新すれば、多様なＤＳＰ関数の呼出記述を、最適化の適用の面で演算と同様に扱うことができるようになる。

【００２９】また、前記所定の関数名の関数は、積和演算及び飽和演算を含むディジタル信号処理用の演算のいずれかを実行することを処理内容とする関数であり、前記プロセッサは、ディジタル信号処理用の演算命令を解読し実行することができるものであり、前記機械語変換手段は、中間コードＢをディジタル信号処理用の演算命令に変換することとすることもできる。また、前記最適化手段によりなされる最適化は、共通部分式の削除、定数伝搬、又はコピー伝搬を含むこととすることもできる。これにより、プログラム作成者は、積和演算、飽和演算等のディジタル信号処理用の演算命令を解読実行可能なプロセッサに対応して、これらの演算を実行させるようなプログラムを作成するに当たり、記述容易性、移植容易性等に鑑みプログラムを高級言語で記述しても、効率の良いコードを得ることができる。即ち、ディジタル信号処理用の演算命令を、所定の関数の呼出の形式で記述することにより、通常の演算と同様に共通部分式の削除、定数伝搬、コピー伝搬等の最適化が行われ、効率の良い機械語命令列に変換されることが保証される。

【図面の簡単な説明】

【図１】ＤＳＰ関数を呼び出す旨の記述を含むソースプログラムの例を示す図である。

【図２】図１に示すソースプログラムに対応してプログラム変換装置により生成される中間コードの例を示す図である。

【図３】本発明の実施の形態に係るプログラム変換装置１００の機能ブロック図である。

【図４】特殊演算対応表１０４の内容を示す図である。

【図５】機械語変換表１０７の内容を示す図である。

【図６】プログラム変換装置１００の動作を示すフローチャートである。

【図７】特殊演算処理部１０３の行うＤＳＰ関数呼出の中間コードの変換処理を示すフローチャートである。

【図８】特殊演算処理部１０３により変換された後の中間コードの例を示す図である。

【図９】図８に示す中間コードに対する中間コード最適化部１０５による処理結果を示す図である。

15

【図１０】図９に示す中間コードを翻訳部１０６で機械語命令列に変換した結果を示す図である。

【図１１】図１に示したソースプログラムをＣ＋＋言語の多重定義演算子を用いて記述した例を示す図である。

【図１２】従来のプログラム変換装置の機能ブロック図である。

【図１３】従来のプログラム変換装置９００における翻訳部９０５の出力の例を示す図である。

【図１４】従来のプログラム変換装置９００におけるＤＳＰ関数展開部９０６の処理結果の例を示す図である。

【符号の説明】

１００　プログラム変換装置
１０１　入力ファイル
１０２　中間コード生成部
＊１０３　特殊演算処理部
１０４　特殊演算対応表
１０５　中間コード最適化部
１０６　翻訳部
１０７　機械語変換表
１０８　出力ファイル
９００　プログラム変換装置
９０１　入力ファイル
９０２　機械語ＤＳＰ関数格納ファイル
９０３　中間コード生成部
９０４　中間コード最適化部
９０５　翻訳部
９０６　ＤＳＰ関数展開部
＊　９０７　出力ファイル

16

【図１】

```
int x;
int y;
int a;

void f(void)
{
        :
        :
   if ( _satadd(x, y) > 0 )
        a = _satadd(x, y);
        :
        :
}
```

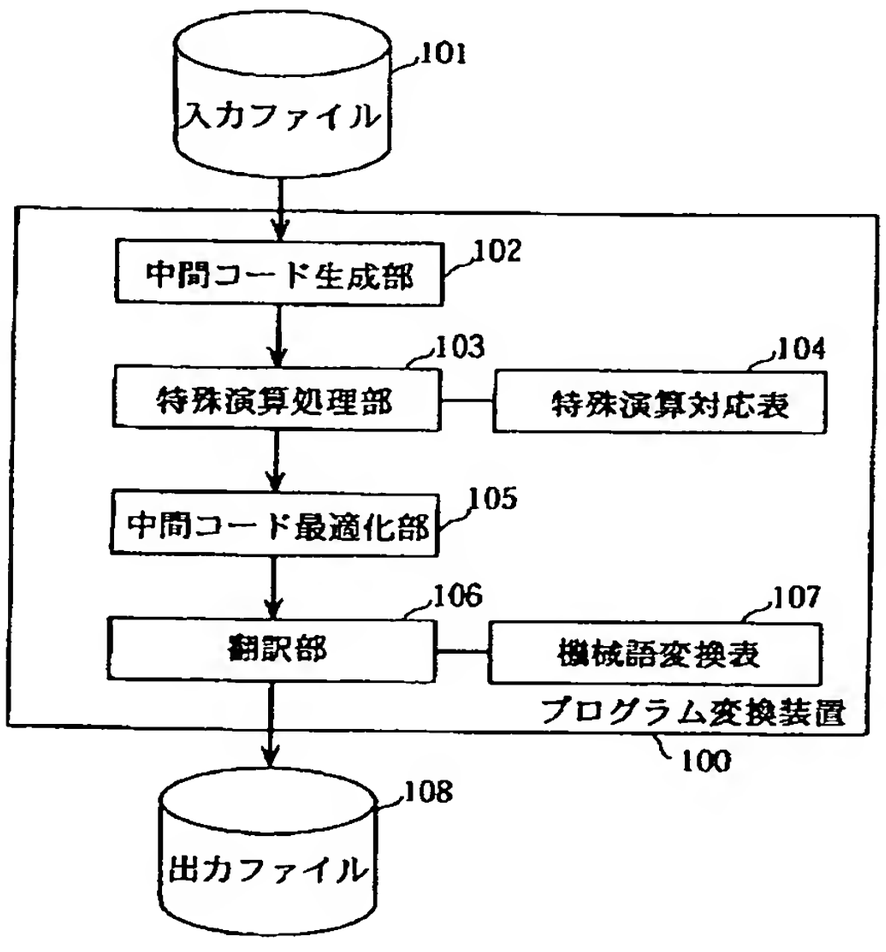
【図２】

```
        :
        :
tmp = _satadd(x, y)
tmp cmpgt 0
jmp_false Label1

a = _satadd(x, y)

Label1
        :
        :
```

【図３】



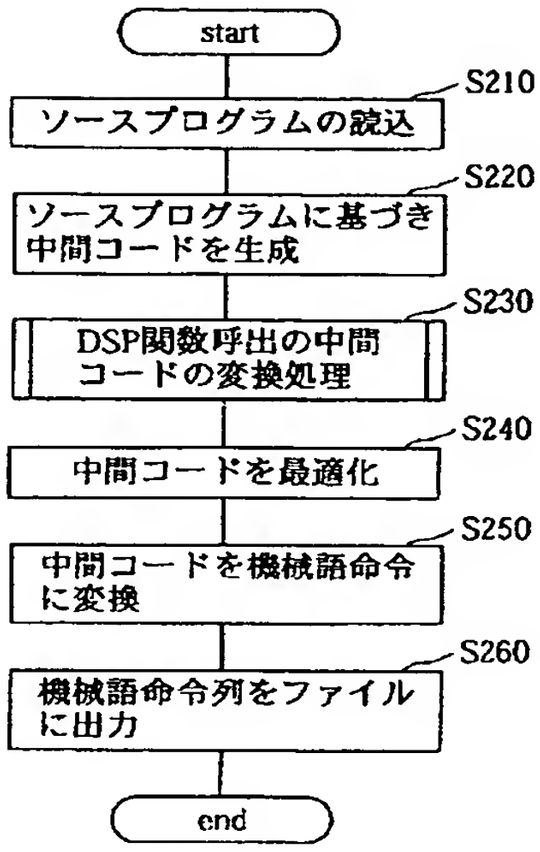
【図４】

DSP関数呼出の中間コード	特殊演算用の中間コード
_satadd(x,y)	x satadd y
_satsub(x,y)	x satsub y
_satmul(x,y)	x satmul y
・ ・ ・ ・ ・	・ ・ ・ ・ ・

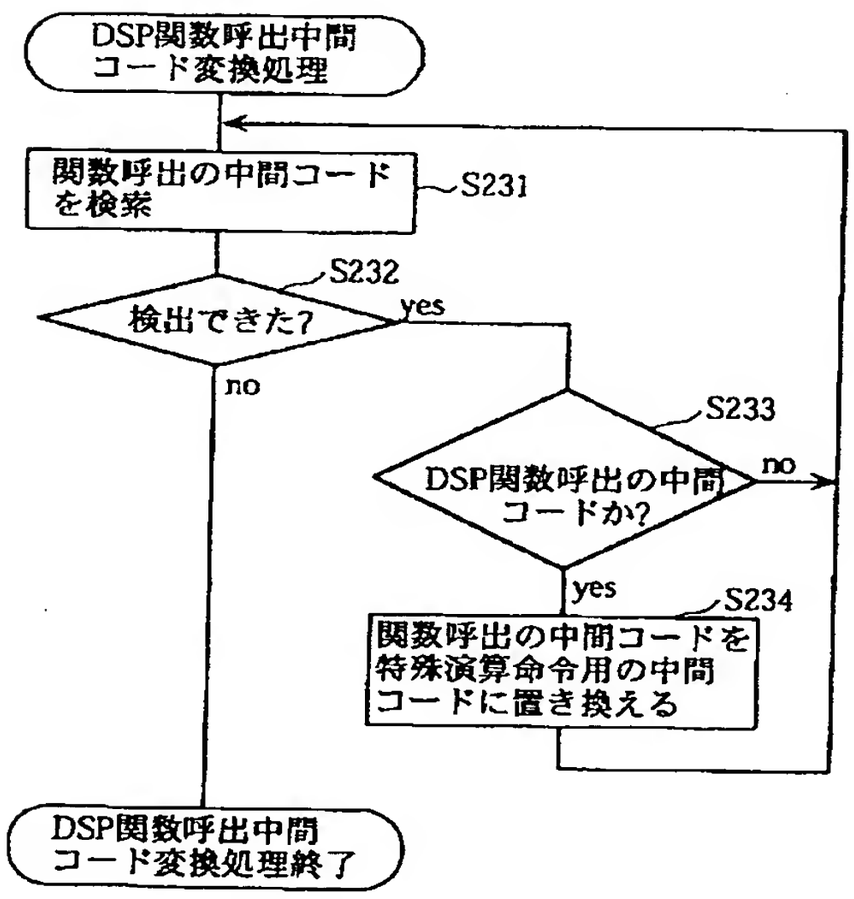
【図５】

中間コード	機械語命令
add	add
sub	sub
・ ・ ・	・ ・ ・
satadd	sadd
satsub	ssub
・ ・ ・	・ ・ ・

【図６】



【図７】



【図８】

```
        :
        :
tmp = x satadd y            ;;飽和加算演算の中間コード
tmp cmpgt 0
jmp_false Label1

a = x satadd y              ;;飽和加算演算の中間コード

Label1
        :
        :
```

【図９】

```
        :
        :
tmp = x satadd y
tmp cmpgt 0
jmp_false Label1

a = tmp                         ;;共通部分式の削除

Label1
        :
        :
```

【図１０】

```
        :
        :
  mov   (x), R3
  mov   (y), R4
  sadd  R3, R4                ;;飽和加算命令
  cmp   R4, 0
  blt   Labl

  mov   R4, (a)

Labl:
        :
        :
```

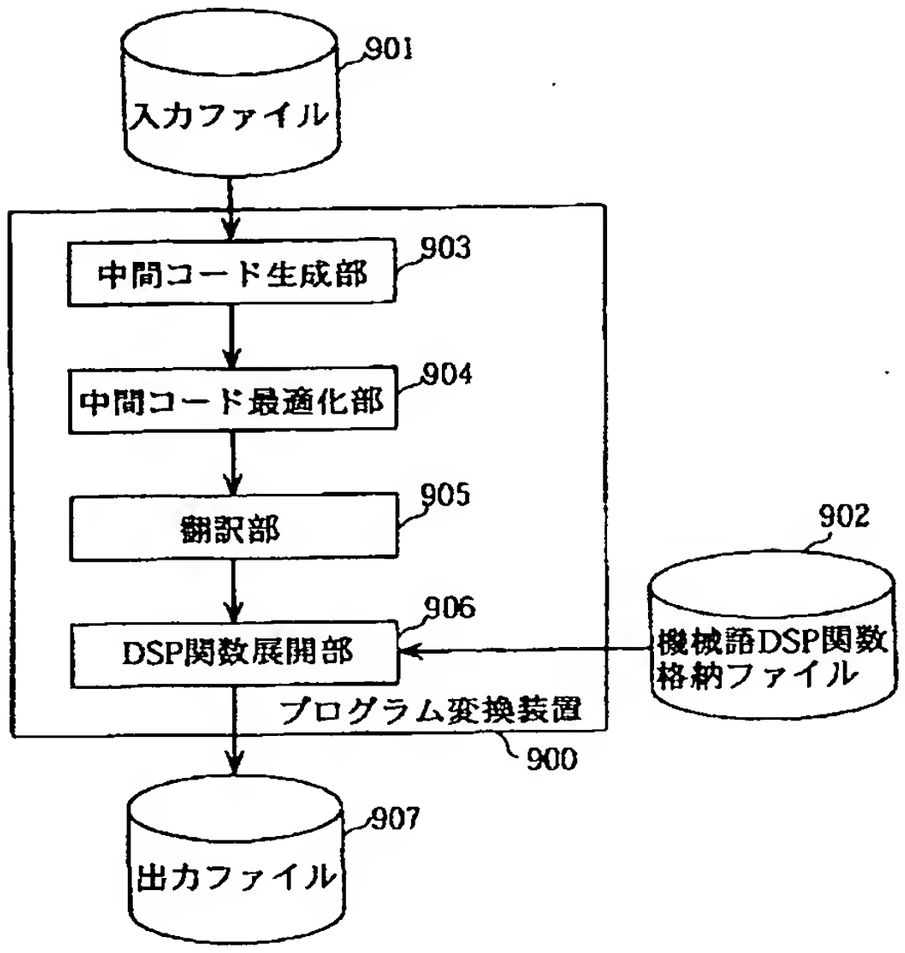
【図１１】

```
/* プログラム変換装置提供者が用意 */
class SatInt {
    int x;
  public:
    SatInt(int);
    SatInt& operator+(SatInt&);          //飽和加算の多重定義演算子
    SatInt& operator-(SatInt&);          //飽和減算の多重定義演算子
    SatInt& operator*(SatInt&);          //飽和乗算の多重定義演算子
        :
};


/* プログラム開発者が記述 */
SatInt x;
SatInt y;
SatInt a;

void f(void)
{
        :
        :
   if ( x + y > 0 )                      //多重定義演算子を使用した記述
        a = x + y;                       //多重定義演算子を使用した記述
        :
        :
}
```

【図１２】



【図１３】

```
        :
        :
mov   R1, R3          ;;R1レジスタの退避
mov   R2, R4          ;;R2レジスタの退避

mov   (x), R1         ;;引数の設定:R1
mov   (y), R2         ;;引数の設定:R2
jsr   _satadd

cmp   R1, 0           ;;戻り値:R1
blt   Lab1

mov   (x), R1         ;;引数の設定:R1
mov   (y), R2         ;;引数の設定:R2
jsr   _satadd

mov   R1, (a)         ;;戻り値:R1

Lab1:
        :
        :
```

【図１４】

```
        :
        :
mov   R1, R3          ;;R1レジスタの退避
mov   R2, R4          ;;R2レジスタの退避

mov   (x), R1
mov   (y), R2
sadd  R2, R1          ;;インライン展開

cmp   R1, 0
blt   Lab1

mov   (x), R1
mov   (y), R2
sadd  R2, R1          ;;インライン展開

mov   R1, (a)

Lab1:
        :
        :
```

フロントページの続き

(72)発明者　漆原　誠一
大阪府門真市大字門真1006番地　松下電器産業株式会社内

(72)発明者　春名　修介
大阪府門真市大字門真1006番地　松下電器産業株式会社内

Ｆターム(参考)　5B081 AA06 AA09 CC16 CC22 CC28 CC41

(11) Publication number : Japanese Patent Laid-Open No. 2000-56981
(43) Date of publication of application : February 25, 2000
(51) Int. CI. : G06F 9/45
(21) Application number : Japanese Patent Application No. 10-222657
(22) Date of filing : August 6, 1998
(71) Applicant : MATSUSHITA ELECTRIC IND CO LTD
(72) Inventor : SAKATA TOSHIYUKI, TOMINAGA NOBUTERU, URUSHIBARA SEICHI, HARUNA NAOSUKE
(54) Title of Invention : PROGRAM TRANSFORMATION DEVICE

SPECIFICATION <EXCERPT>

[0002]
[Prior Art] In recent years, in order to perform multimedia processing such as image processing at a high speed, instructions for digital signal processing (hereinafter, referred to as DSP instructions), such as a product-sum operation and a saturate calculation, have been in microprocessors. The software used in these microprocessors is usually created by translating a source program described in a high level language, such as C language and object-oriented language C++ (hereinafter, referred to simply as "C++ language"), into machine language instructions using a program transformation device. However, since the above-mentioned high level language is not for the DSP instructions, the language cannot describe operators representing the DSP instructions. Therefore, conventionally, in order to use the DSP instructions, functions corresponding to the DSP instructions (hereinafter, referred to as DSP functions) have been generated in a machine language, and calls for the DSP functions in a source program have been described in high level language. Since

description of calls for functions can be described in a high level language, by describing the calls for the DSP functions in the source program in a standardized high level language, it is possible to unify source programs corresponding to various processors.

[0003] The following describes processing in which the conventional program transformation device transforms a source program shown in FIG. 1 into a machine-language instruction sequence. FIG. 1 is a diagram showing one example of a source program including description of calls for DSP functions. The source program shown in FIG. 1 is described in the C language, and "_satadd (x y)" is a description of a call for a DSP function. Note that this DSP function is a function which performs a saturate addition operation. FIG. 12 is a functional block diagram of the conventional program transformation device. The input file 901 stores the source program having the description of the calls for the DSP functions. A program transformation device 900 is a so-called compiler which reads the source program stored in the input file 901, translates the source program into a machine-language instruction sequence, and outputs the sequence to an output file 907. The program transformation device 900 includes an intermediate code generation unit 903, an intermediate code optimization unit 904, a translation unit 905, and a DSP function expansion unit 906. The intermediate code generation unit 903 transforms the source program that has been read from the input file 901 into an intermediate code that is an internal representation to be used in the program transformation device. As a result of this transformation of the source program of FIG. 1 by the intermediate code generation units 903, an intermediate code shown in FIG. 2 is generated. FIG. 2 is a diagram showing one example of the intermediate code generated by the program transformation device corresponding to the source program shown in FIG. 1. For example, "tmp=_satadd (x y)" and "tmp cmpgt 0" in the intermediate codes shown in FIG. 1 correspond

to "_satadd(x y) >0" in the source program shown in FIG. 1.

[0004] The intermediate code optimization unit 904 optimizes inline expansion of a function, deletion of a common subexpression, constant propagation, copy propagation, and the like for the intermediate code generated by the intermediate code generation unit 903. However, since "_satadd (x y)" in the intermediate code shown in FIG. 2 represents a call for a DSP function, although there are a plurality of the same descriptions, it cannot be treated as common subexpression and the intermediate code optimization unit 904 does not delete a common subexpression. Here, the inline expansion of a function replaces a part for calling the function with the intermediate code indicating detail of a function, which make it possible to eliminate subroutine linkage and instructions for the subroutine linkage. If subexpressions are evaluated to have an equivalent, the deletion of a common subexpression stores a result of each estimation as a variable, and replaces a next common subexpression with the variable, thereby enables a plurality of evaluation processes for equivalent subexpressions to be performed only once. After a substitution statement v=a which substitutes, for example, a constant a for a variable v, the constant propagation replaces a reference to v with the constant a until the substitution to the variable v appears, thereby calculating the part during translation when the variable in the statement can be replaced with a constant, so that the original statement can be replaced with the calculated result, for example. Moreover, the copy propagation replaces a constant in constant propagation with a variable. In addition, operators and operands in the source program described in the high level language, such as "x+y" and "x-y", serve as an object which is applied with optimization of deletion of a common subexpression, constant propagation, copy propagation, and the like, even after being transformed into a intermediate code. The translation unit 905 translates the intermediate code optimized by

the intermediate code optimization unit 904 into machine-language instruction. The translated result is shown in FIG. 13.

[0005] FIG. 13 is a diagram showing one example of the output from the translation unit 905 in the conventional program transformation device 900. The DSP function expansion unit 906 detects a subroutine call instruction which calls a DSP function in the machine language instruction translated by the translation unit 905, develops the subroutine call instruction into the DSP function which is the machine-language instruction sequence stored in a machine-language DSP function storage file 902, and outputs the result to the output file 907. The processing result of the DSP function expansion unit 906 is shown in FIG. 14. FIG. 14 is a diagram showing one example of the processing result of the DSP function expansion unit 906 in the conventional program transformation device 900. In addition, in FIG. 14, the machine language instruction sequence is expressed as an assembly language program for convenience. Thus, the conventional program transformation device 900 transforms a part of the source program shown in FIG. 1 to the machine-language instruction stream shown in FIG. 14.